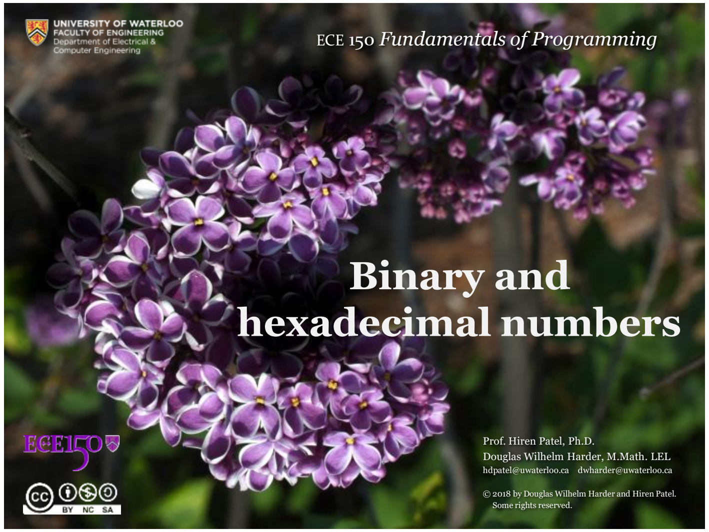


Outline

- In this lesson, we will:
  - Learn about the binary numbers (bits) 0 and 1
  - See that we can represent numbers in binary
  - Quickly introduce binary arithmetic and multiplication
    - It is completely parallel to decimal addition and subtraction
  - Consider a more compact representation: hexadecimal
    - The translation between binary and hexadecimal uses a very simple *look-up table*

Counting

- We count in base 10 (called *decimal* counting), meaning we have 10 unique digits, and then we use a *positional number system* to represent larger numbers
  - Once we get to the largest number in any position, we increment the next highest unit
- Base 10 is really only useful for humans: we have ten fingers
  - Our clock, however, is a hybrid:
    - There are 60 seconds in a minute
    - There are 60 seconds in an hour
    - There are 24 hours in a day
  - You know that
    - One second after 23:59:59 is 0:00:00 the next day
    - The next highest number after 999 is 1000

Counting

- Base 10 is great for humans: we have five fingers on each hand
- It's more difficult for computers:
  - Numbers are stored as voltages
    - If you want ten different voltages representing ten different digits, you must recognize and store these voltages—this is very difficult
  - It is easier to store, access and manipulate just two voltages:
    - Say 0 V and 5 V
  - This leaves us with two digits only, say `0` and `1`
  - We will describe `0` and `1` as ***bi**nary digi**ts*** or *bits*

Counting in binary

- If we only accept two bits (0 and 1, or 0 V and 5 V), it may seem much worse, but it's still manageable:

0b0
0b1
0b10
0b11
0b100
0b101
0b110
0b111
0b1000
0b1001

Counting in binary

- Question: Is 100 equal to 10^2 or 4?
  - We will usually:
    - Prefix binary numbers with "0b"
    - Use the monospaced typeface Consolas
  - Thus:
    - 100011010 is a large decimal number
    - 0b11110110010 is binary for 1970
  - To start, we will gray-out the "0b"

Counting in binary

- These first ten non-zero binary numbers could therefore represent the number of "I"s shown here

0b0	
0b1	I
0b10	II
0b11	III
0b100	IIII
0b101	IIIII
0b110	IIIIII
0b111	IIIIIII
0b1000	IIIIIIII
0b1001	IIIIIIIII
0b1010	IIIIIIIIII
0b1011	IIIIIIIIIII

Counting in binary

- Normally, however, we just indicate the decimal number that it is equivalent to

0b0	0
0b1	1
0b10	2
0b11	3
0b100	4
0b101	5
0b110	6
0b111	7
0b1000	8
0b1001	9
0b1010	10
0b1011	11

Counting in binary

- In decimal, each successive digit represents a higher power of ten:

10	10^1
100	10^2
1000	10^3
10000	10^4

- In binary, each successive digit represents a higher power of two:

`0b10`	$2^1 = 2$
`0b100`	$2^2 = 4$
`0b1000`	$2^3 = 8$
`0b10000`	$2^4 = 16$

Counting in binary

There are only 10 types of people in the world.

Those who understand binary, and those who do not.

Binary integers

- For example, in decimal, the number

  280973

  represents

  $2 \times 10^5 + 8 \times 10^4 + 0 \times 10^3 + 9 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$

- In binary, the number

  `0b101010`

  represents

  $1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 42$

Binary real numbers

- Similarly,

  280.973 Uses a "decimal point"

  represents

  $2 \times 10^2 + 8 \times 10^1 + 0 \times 10^0 + 9 \times 10^{-1} + 7 \times 10^{-2} + 3 \times 10^{-3}$

- In binary, the number

  `0b101.010` Uses a "radix point"

  represents

  $1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$

  $4 + 1 + 0.25 = 5.25$

Note: Every decimal integer is followed by an implied decimal point
Every binary integer is followed by an implied radix point

Addition

- Just like addition with decimal numbers, you can do the same with binary, you only have to remember:

`1 + 1 = 10` and `1 + 1 + 1 = 11`

```
  1 1 1                 1 1 1 1
   3725                  1010
 + 8982              +    111
  12707                 10001
```

```
 1 1   1   1 1 1     1 1          1 1 1 1 1    1    1 1     1 1
          9275948135782                1110010001100111
+ 5032925823853322553           + 11011101101100110
  5033018583334583335             100010110110010110 1
```

```
          9275948135782               1110010001100111
+ 5032925823853322553         + 11011101101100 0110
```


Multiplication

- Just like multiplication with decimal numbers, you perform the same operations here, too, only it is easier, just more tedious

```
                                                      1100101110000010
                                                    × 101000101
            359801         111011                     1100101110000010
   42     ×   4327       ×   1011                    00000000000000000
 × 61      2518607         111011                  110010111000001000
   42      7196020        1110110                 0000000000000000000
+ 2520    107940300      000000000               00000000000000000000
 2562   + 1439204000   + 111011000              000000000000000000000
          1556858927     1010001001           1100101110000010000000
                                             00000000000000000000000
                                         + 110010111000001000000000
                                           1000000100101110000001010
```


Note

- We will not be performing difficult addition or multiplication problems in binary
  - The computer does these calculations and conversions
  - They will be reduced to adding `1` or multiplying by `10`

Verbosity

- One weakness of binary numbers is that they are verbose:
  - A binary number representing the same quantity as the corresponding decimal number will have approximately 3.322 times as many digits

999
`0b1111100111`

2^{10} = 1024
`0b10000000000`

10659860817235789230
`0b1001001111101111011011101110111101010110011100011100010110101110`

2^{100} – 1 = 1267650600228229401496703205375
`0b1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111`

- We will need binary numbers, because that is how almost everything in the computer is stored
  - Unfortunately, it's very difficult to convert from binary to decimal and vice versa… ☹

4-bit translation

- Instead, note that there are 16 different quadruple of binary digits:

0b0000	0
0b0001	1
0b0010	2
0b0011	3
0b0100	4
0b0101	5
0b0110	6
0b0111	7
0b1000	8
0b1001	9
0b1010	10
0b1011	11
0b1100	12
0b1101	13
0b1110	14
0b1111	15

4-bit translation

- Represent every quadruple with a unique digit:

0b0000	0
0b0001	1
0b0010	2
0b0011	3
0b0100	4
0b0101	5
0b0110	6
0b0111	7
0b1000	8
0b1001	9
0b1010	a
0b1011	b
0b1100	c
0b1101	d
0b1110	e
0b1111	f

This is called a *look-up table*

We ran out of digits...
...we could pick digits
from another language,
say Arabic?

4-bit translation

- Always start by grouping binary digits around the radix point and add extra zeros at the start to make a multiple of 4 binary digits:

0b0000	0
0b0001	1
0b0010	2
0b0011	3
0b0100	4
0b0101	5
0b0110	6
0b0111	7
0b1000	8
0b1001	9
0b1010	a
0b1011	b
0b1100	c
0b1101	d
0b1110	e
0b1111	f

```
0b00010010101001001000100110110111
   1   2   a   4   8   9   b   7
```

This binary number is represented by `0x12a489b7`
– We will prefix this with `"0x"`

4-bit translation

- To go the other way, just replace of our 4-bit translations by the corresponding quadruple:

0b0000	0
0b0001	1
0b0010	2
0b0011	3
0b0100	4
0b0101	5
0b0110	6
0b0111	7
0b1000	8
0b1001	9
0b1010	a
0b1011	b
0b1100	c
0b1101	d
0b1110	e
0b1111	f

```
   6   c   f   0   d   5   3   e
0b01101100111100001101010100111110
```

This 4-bit translation represents
`0b1101100111100001101010100111110`

– We stripped off the leading `0`

Hexadecimal numbers

- What we are actually doing is representing the numbers in base 16
  - This is called *hexadecimal* (base six-and-ten)
  - Often abbreviated as *"hex"*
  - This is where the "x" comes form in `"0x"`
  - Again just like binary uses two digits: `0` and `1`
    decimal uses ten digits: 0 through 9
    hexadecimal uses sixteen digits `0` through `f`
- It is used for nothing more than a compact representation of binary
  - Conversion between decimal and hex and vice versa is difficult
  - To convert between binary and hex is easy
    - Just remember to always start at the radix point

Hexadecimal numbers

- What you really need to know for this course: `9 + 1 = a`, `f + 1 = 10`
  - These are consecutive hexadecimal numbers

```
0x27a932f8    0x27a93307
0x27a932f9    0x27a93308
0x27a932fa    0x27a93309
0x27a932fb    0x27a9330a
0x27a932fc    0x27a9330b
0x27a932fd    0x27a9330c
0x27a932fe    0x27a9330d
0x27a932ff    0x27a9330e
0x27a93300    0x27a9330f
0x27a93301    0x27a93310
0x27a93302    0x27a93311
0x27a93303    0x27a93312
0x27a93304    0x27a93313
0x27a93305    0x27a93314
0x27a93306    0x27a93315
0x27a93307    0x27a93316
```


Summary

- Following this lesson, you now
  - Understand that computer use binary numbers
  - Know that the digits `0` and `1` are called bits
    - Binary numbers are prefixed by `"0b"`
  - See that binary addition and multiplication mirrors decimal addition and multiplication
  - Understand that binary numbers are verbose
    and hexadecimal representations are more compact
    - Hexadecimal numbers are prefixed by `"0x"`
  - Know how to translate between binary and hexadecimal and back
    - You don't care what decimal value a hexadecimal number is…

References

[1] Wikipedia:
https://en.wikipedia.org/wiki/Binary_number
https://en.wikipedia.org/wiki/Hexadecimal
https://simple.wikipedia.org/wiki/Hexadecimal_numeral_system

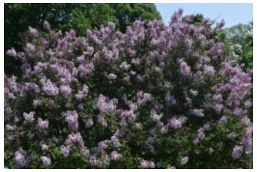
Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.


Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.